Itinerant Delivery of Popular Data via WIDE Hot Spots*

Sinan Isik, Mehmet Yunus Donmez and Cem Ersoy

Department of Computer Engineering, Bogazici University,
Istanbul, Turkiye
{isiks, donmezme, ersoy}@boun.edu.tr

Abstract – Wireless Information Delivery Environment (WIDE) is a distributed data dissemination system, which uses IEEE 802.11b technology. WIDE aims to deliver popular information services to registered mobile clients in WLAN hot spots. Data delivery is based on broadcasting and multicasting to provide scalability and efficient use of the wireless channel. Reliability is assured with a combination of Forward Error Correction (FEC), data carousel, and ARQ techniques. This paper presents the proposed system architecture with the details of reliable and secure data dissemination mechanisms. Functional evaluation of the proposed system and mechanisms on the implemented prototype are also included in this paper.

Index Terms – reliable and secure data dissemination, wireless hot spots, wireless LANs

1. Introduction

The dominant use of WLANs is to provide Internet access for mobile users in distributed hot spots. Users can search for and access the information over the Internet with the help of their wireless capable PDAs or laptop computers even while they are itinerant. However, there are some scenarios where there is highly correlated demand for specific information, which we refer as popular data. Course notes, web pages and announcements for students in a campus environment; detailed information about the objects for visitors in an exhibition; or product and price information and promotions for customers in a shopping-mall can be regarded as popular

* This work is partially supported by the State Planning Organization of Turkey under the grant number 98K120890, and by the Bogazici University Research Projects under the grant number 02A105D. A shorter version of this paper was presented in WONS 2004 in Madonna di Campiglio, Italy.

data. The content of these data may change in time depending on the subject of the data, as in the cases of the content of a web page or a news service. The aim of WIDE system is to bring such popular information ready to the fingertips of WLAN users rather than make them search for it.

The WIDE system can be defined as a collection of hot spots providing high bandwidth radio link for data services in *moderate-sized* and *low-mobility (walk-through)* environments. The system takes on the concept of discontinuous service provision because of the non-ubiquitous nature of hot spots. For many types of information, discontinuous coverage is sufficient. Email, music, or e-book services can be classified within this category.

WIDE hot spots can be found in locations where there is the appropriate user density, but users have to walk through the service area in order to actually access the service, and then take their service away for later consumption. As users pass through the coverage area of the system, the most recent version of the information services in the user profile will be automatically downloaded to their mobile terminals without any user intervention. A typical WIDE client and its user interface are illustrated in Figure 1.

(a) WIDE client (b) User interface

Figure 1. A typical PDA client and its user interface

Interesting deployment scenarios are possible for large-scale networks. WIDE hot spots located at traffic posts and/or bus stops may provide municipal services to both pedestrians and

to the cars waiting for the red light or for the bus. Users of the system may subscribe to the information services related to that region, including local maps, weather reports, shop, and restaurant locations, as well as road conditions and traffic congestion reports.

The motivation of the system proposed in this paper is the Infostation concept [1,2]. Hot spots of the WIDE system are analogous to Infostation cells in that both offer discontinuous service provision. There are ongoing projects on the Infostation concept. One of these projects focuses primarily on the data link layer and below to enable drive-through data reception [3] and another one focuses on developing Infostation applications and transport layer mechanisms to support those applications for walk-through scenarios [4,5]. Rover Technology [6] and Mobius [7] are other ongoing projects specialized on employing location tracking to decide the services to deliver to the users, and scheduling algorithms for large-scale data dissemination systems respectively. Wideray [8] is a commercial product providing some of the functionality of WIDE. However, Wideray is based on GSM networks, not on distributed WLAN hotspots and uses a hardware called Jack utilizing IrDA to provide service requiring user interaction.

In Section 2, we give a brief overview of the design principles to implement the WIDE system. Section 3 presents the system architecture and communication design of WIDE. The details of the mechanisms included in WIDE are presented in Section 4. Section 5 presents the functional evaluation of the proposed system design on the implemented WIDE prototype. Section 6 concludes the paper, suggesting some future works.

2. Design Principles of WIDE

There are some basic requirements that should be considered in the design of the WIDE system, which are necessary for the dissemination of popular data in a secure and reliable way, requiring little or no human-computer interaction.

The WIDE system only gives service to its registered subscribers. Hence, clients have to be authenticated in a secure way by the system in each visited hot spot to be able to receive any services. Additionally, a global security mechanism should be designed so that the network packets of WIDE system could only be identified and processed within components of WIDE.

A publish/subscribe mechanism should be designed to store the preferences (i.e. subscriptions) of the subscribers in their user profiles. In that way, the transfer of any information services and their updates can be arranged in a way that requires little or no human-computer interaction, and can be completed as clients pass through the service area of a server. Specifically, in a walk-through scenario, assuming a walking speed of 2 m/s, the residence time of a user traveling along the 150 m diameter of a hotspot is approximately one minute.

Since the system offers popular information services, it should perform at an acceptable level in terms of reception time for individual clients when there are many users demanding the same service. Hence, the design should be based on data broadcasting, or, more precisely data multicasting to provide scalability and efficient use of the wireless channel.

The system protocols have to be designed to conserve energy. Energy conservation can be achieved by dozing the Wi-Fi card at every unnecessary communication period. However, the discussion and implementation of this issue is deferred to a follow up study.

Provided protocols must satisfy the reliability needs of the wireless medium. The residence time of a client in the coverage area may be very short which may lead to an incomplete data transfer. The completion of any incomplete data transfer should be dealt with by the system infrastructure using some recovery and error correction mechanisms. In addition, the protocols for data transfer should be designed in a way that allows the coexistence of other communication traffic on the wireless channel.

Additionally, WIDE system should operate at the transport layer and above using existing protocols without requiring any modifications on the MAC and Data Link Layer protocols and should be compatible with TCP/IP to enable easy deployment of the system. Hence, WIDE design should be independent from the underlying network infrastructure.

3. Wireless Information Delivery Environment

WIDE is a system, which is designed for delivering popular information services in moderate-sized environments. There are components having specific tasks in WIDE design, which form the WIDE network. The network is composed of both wired and wireless mediums. Two main system components, called WIDE Cluster Controller (WICC) and WIDE Servers (WIS) are located in the wired part of the system and they communicate through WIDE LAN. The building blocks of the WIDE system and their connectivity are shown in Figure 2.

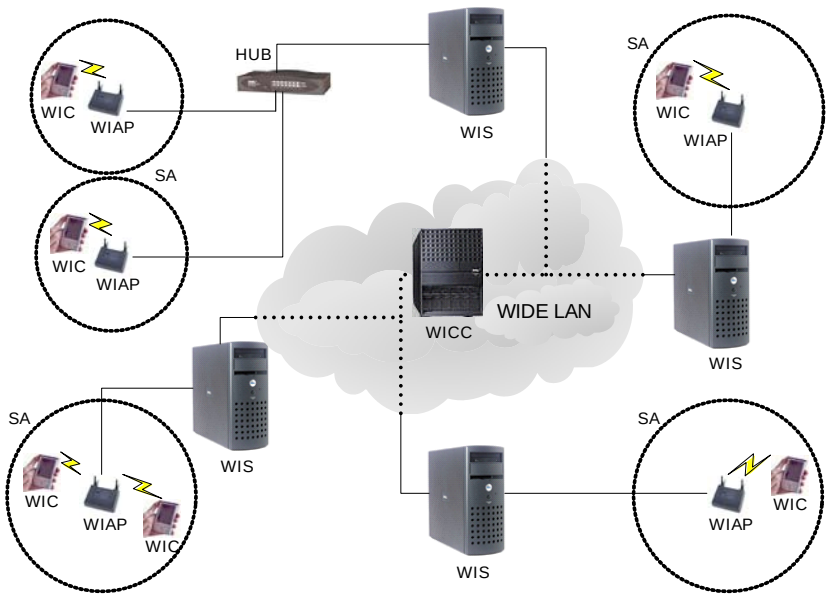

Figure 2. Building blocks of WIDE

WIDE Servers are responsible for preparing and delivering information services to clients. The information services, which are available for delivery to clients, are assumed to be

stored on each WIS. The delivery management information such as service identifier, class, version, name and location on the local disk is recorded in WIDE Server Database (SDB).

WIDE Cluster Controller keeps and manages system administration database called WIDE Cluster Controller Database (CCDB). Information about servers, user profiles, services offered by the system and authentication of clients are stored in this database.

The IEEE 802.11b WAPs, namely WIDE Access Points (WIAPs) are located at the end-points of the wired network to provide wireless connectivity of WIDE Clients (WIC) to the servers of the system, forwarding traffic in both directions. A client is a battery operated handheld or laptop PC with necessary equipments that provide wireless connectivity.

There can be one or more WIAPs connected to a server, but a WIAP can only be connected to one server. The geographical area covered by WIAPs that are connected to a server is called the Service Area (SA) of that server.

3.1. Network Protocols Used in WIDE

The communication of the servers and the clients is established using IP protocol. Servers of the system may be located in different subnets of the WIDE network. The IP address of a client, which is valid in a service area, may not be valid in another one. An automatic address configuration scheme is needed to provide continuous communication.

WIDE is designed to operate independent from how the IP level connectivity of clients and servers is established. IP addressing of clients can be established using DHCP [9], where WIDE servers are configured as DHCP servers. Instead of DHCP, a faster protocol such as DRCP [10] may be employed. Mobile IP [11] can also be employed by configuring servers as a foreign agent. The care-of address assigned by foreign agent will enable clients to receive service from the system. In IPv6 [12], stateless autoconfiguration solves the address configuration

problem of clients. In the prototype, DHCP is used for the automatic address configuration, which is a default component of TCP/IP protocol in Windows platform.

In the transport layer, WIDE uses UDP in client-server communication. Since TCP does not support broadcasting and multicasting, the system cannot deliver popular data to multiple users simultaneously using TCP. If a large number of clients are connected to the server and receiving information services from that server, contention for the shared wireless channel to send NACKs and ACKs will waste additional time. Establishing a connection is another bottleneck for clients because the overhead required for it will take time and energy [5]. In the communication between server and cluster controller, TCP protocol is used.

3.2. Data Delivery Cycles of WIDE

Client-server communication is divided into periodic time frames, which are also divided into sub-frames. Each sub-frame is a time period, which is dedicated to a certain communication task. This frame structure is formed to organize the message and data transfers. Since there is only one physical channel, namely the wireless medium, such an organization is needed to decrease the number of collisions and to provide efficient use of the channel. Periodic time frames are called Communication Cycles (CCs). Figure 3 illustrates the structure of the CC.

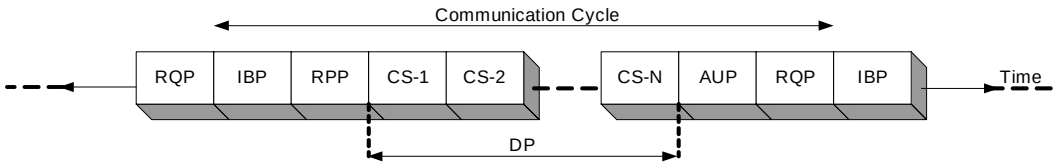

Figure 3. Communication Cycle

The sub-frames, which are named as index broadcast period (IBP), reception preparation period (RPP), data period (DP), authentication period (AUP) and request period (RQP), sequentially follow each other in this order in a CC. DP is also divided into sub-frames, called

communication slots (CS). The duration of DP is dependent on the number of utilized CSs, whereas the other periods are of fixed size.

CC design is independent from the underlying MAC layer. The collisions and errors at the MAC layer are perceived as delayed packets at the application layer i.e., packets may arrive after the related sub-frame passes. Hence, CC is not strictly time sensitive and this does not disturb the operation of the CC in our design. The system uses its error recovery mechanisms at the application layer to deal with the packet losses.

In the execution of a typical communication cycle, clients entering the service area listen to the broadcast messages initiated by that server. Clients should authenticate themselves to the system in order to receive service from that server. The authentication messages and their responses are sent during the AUPs. In the current implementation, a client reinitiates its authentication after two cycles, the reason of which will be clarified in the experiments section.

The subscription and unsubscription requests of clients and the response of the server to these messages are sent during the same RQP. In addition, retransmission requests for incompletely received information services and polling requests for updates on the user profile are also transmitted during the RQP.

A scheduler running in the server decides the data to transmit during each CC and prepares the index. The scheduling of an information service in a server requires at least one subscribed client in the service area. If any client has just subscribed to that information service or a retransmission is requested for that service due to incomplete reception, then that service is queued for delivery. Moreover, if any client has made an authentication or polling request and if there is a newer version of an information service in the user profile of any client, it is queued for delivery. At the time of delivery, the service appears in the index.

Clients are informed about the information services offered by the server during that CC, by examining the index message received during the IBP. Index message informs the clients about the multicast group of transmission, the version and the size of the information service to be transmitted. Each multicast group is coupled with a CS in DP. The client determines whether there are any available items of interest based on the user profile in the mobile terminal. Clients interested in the services announced in the index join the announced multicast groups and prepare the buffers for reception during the RPP. Services are delivered to clients in the form of packets of fixed size to keep the error chance equal for all packets. In the current implementation the size of data packets is configured to be 1400 bytes, which is explained in Section 5.1.

Data packets of each item announced for that CC are delivered in the corresponding CS. Clients receive the packets of the interested service by listening to the relevant multicast group.

The choice of synchronous communication cycle is related with performance and power-awareness issues. We aim to deliver high throughput data service since the residence time of a user in the service area may be very short. We provide performance optimization by partitioning the time into periods of certain tasks. In this way, we prevent the server to deal with other requests, which decreases the performance during the data period.

The power-awareness of the clients can be provided with the help of this synchronous scheme, such that clients can doze their Wi-Fi cards if the client is not interested in the data delivered in that CC, by inspecting the index received at the beginning of the CC. The synchronous structure helps the client to guess the approximate arrival time of the next index, since the length of the data period can be calculated by inspecting the current index [13].

3.3. Virtual Channels of WIDE

The sub-frames in a CC utilize *virtual channels*, which are based on UDP broadcast,

multicast, and unicast. Client - server communication is established through these virtual channels. These channels are defined in the transport layer and are identified by their corresponding IP addresses and port numbers. The relationship between virtual channels, time frames and the messages between clients and a server is illustrated in Figure 4.

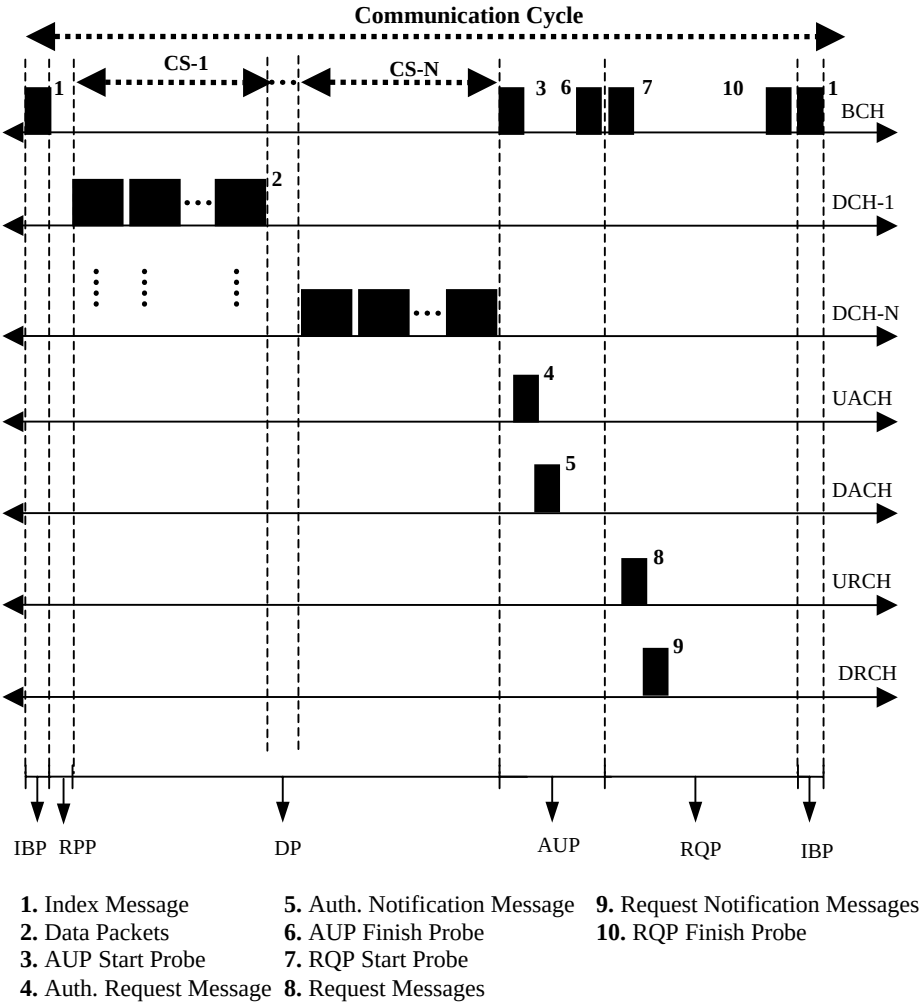

Figure 4. Communication cycle and virtual channels

Broadcast channel (BCH) is the control channel of WIDE. A server sends the index message, and start/finish probes for AUP and RQP periods. When a client enters to the service area of a server, it is informed about the existence of that server and then synchronizes with the communication cycle of that server by listening to BCH.

Uplink/downlink authentication channels (UACH, DACH) are unicast channels used for authentication of clients. Authentication requests are sent over UACH during the AUP and authentication notification messages are sent back over DACH during a following AUP.

Uplink/downlink request channels (URCH, DRCH) are unicast channels used for subscription, unsubscription, retransmission requests and notification purposes. These requests and periodic polling requests are sent over URCH during the RQP. Servers reply with the corresponding notifications of these requests during the same RQP.

Data Channels (DCH) are multicast channels used to deliver information services to clients on the corresponding CS during the DP. Clients are informed about the DCH of the information service by the index message.

4. Reliable and Secure Data Dissemination in WIDE System

4.1. Dissemination of Popular Data

Main characteristic of data dissemination mechanism in WIDE is that the initiation of data delivery is both pull and push based. A pure push mechanism is not used because sending irrelevant data to clients is a waste of resources and more seriously, in the absence of requests, it is possible that specific data that are needed by the clients will not be delivered in a timely fashion. On the other hand, usage of pure pull mechanism results in an explosion of client requests on the server side, interrupting the server continuously to deal with these requests, since in this mechanism, each client has to explicitly request for any updates of each subscribed information service. A pull mechanism is used to enable information delivery only when an interested client is in or has just entered the service area by the help of periodic polling or authentication polling mechanisms respectively. In this way, other communication traffic can

coexist with our system on the wireless channel since the CC is idle in the absence of clients requesting service in the service area.

Whenever a client enters the coverage area of a server, it sends an authentication request. If it is authenticated, then this request acts as a request for polling the updates. To be able to poll data related to this client, a user profile is created by using a publish/subscribe mechanism. Clients subscribe to interested information services to form their user profiles. Each subscription behaves as a pull request for the relevant service. If any update of any service in the user profile exists for the user that requests for authentication, the updated information is delivered to a multicast group without any explicit request after the authentication phase. In addition to authentication polling, periodic polling is used to poll for any updates in the case of a long residence time of clients in the service area. The push behavior of the system is observed as a result of these polling mechanisms. If there are other clients in the service area, who did not receive this updated information, they will also receive that service simultaneously.

4.2. Creation of User Profiles

Preferences of clients, namely subscriptions, are stored in their user profiles with the help of a publish/subscribe mechanism in WIDE. The list of information services offered by the system, namely Table of Contents (TOC), is published to the clients as an information service. The TOC service is in the user profile of each client by default and it is updated in case of any changes in the offered service list. The user profile of a client is stored both in the cluster controller to be used for polling operations, and the client device to keep a list of subscriptions. A user interface is provided to the clients as shown in Figure 1.b to display the TOC and the services in the user profile.

Users may create a subscription and unsubscription requests with the help of the user

interface even when the client is not in any service area. Subscription and unsubscription requests are transmitted automatically to the cluster controller via the server of the service area without any user interaction. The entry for the service that the user wants to unsubscribe from is deleted from the corresponding user profile.

4.3. Reliable Data Dissemination in WIDE Hot Spots

Wireless data delivery systems require reliability mechanisms to ensure that all the intended recipients of an information service receive the service intact. Data carousel or broadcast disk approach [14] can ensure full reliability by delivering the data multiple times. However, this increases the number of unnecessary receptions caused by duplicate packets on the client side. Also, carousel mechanisms increase the average reception time of an information service via unreliable media. Hence, the number of carousel cycles for each information service should be limited. Erasure code mechanisms [15] can be employed to decrease the reception time. Here, the sender prevents losses by transmitting some amount of redundant information, which allows the reconstruction of missing data at the receiver without further interactions. Besides reducing the time needed to recover the missing packets, such an approach generally simplifies both the sender and the receiver since it may render a feedback channel unnecessary. The technique is attractive also for multicast applications, since different loss patterns can be recovered by using the same set of transmitted data. In a multicast environment, an ARQ mechanism might be highly inefficient because of uncorrelated losses at different groups of receivers. Thus, it may be preferable to use ARQ in the cases when the FEC mechanism is not sufficient to reconstruct the data [16].

The reliability of data packets is provided using a mixed type of carousel, erasure code, and ARQ techniques. Before the transmission, data packets of an information service are

encoded using a FEC technique in which the reception of any k packets out of $k+m$ transmitted packets is sufficient for reliable reception [15]. After this phase, a packet number is given to each packet. Clients keep track of the packets and discard the duplicates caused by the carousel mechanism. When enough packets are received for the FEC decoding process, they are decoded to form the actual data. If the number of received packets is not enough to recover the actual data, the missing packets can be captured in the next carousel cycle, if exists. If there are still missing packets to be captured, then an ARQ request is prepared to request the retransmission of that information service.

For the data and authentication request messages initiated by the clients, an acknowledgement scheme is used. The functionality of the system depends on the reliable delivery of these messages to servers. If notifications for data requests are not received by the client in the same request period, then it repeats its requests in the next request period. Similarly, if notifications for authentication requests are not received by the client in two cycles, then the request is repeated in the next authentication period after the expiration of the duration. The broadcast messages announced to the clients do not need notifications, since the loss of these packets due to error conditions does not violate the integrity and the functionality of the client. However, these losses may increase the average reception time of information services, since their losses postpone the invocation of related modules to a subsequent period.

4.4. Secure Data Communication in WIDE

Since mobile terminals, especially handheld computers, have limited battery capacity, the security schemes used in WIDE should be power efficient. Symmetric-key encryption schemes have low complexity and high data throughput, providing fast and power-efficient processing [17]. Therefore, these encryption schemes are used for the user authentication, data origin

authentication, data privacy, and data integrity operations. In addition, since we have a special communication structure between the server and the client, there is no proper authentication scheme, which obeys the timing restrictions of the proposed design. Hence, we concentrated on a system specific security mechanism between the client and the server. Since there is no special communication structure between the servers and the controller, any security scheme such as Kerberos may be used for authentication and authorization purposes.

Each packet exchanged between a server and a client has to be encrypted with a key, which is only known by the endpoints of the communication and the controller, to keep the contents secret from the public. There are three specific keys in the system, which are common key, service key and client key. The common key is known by all of the components of WIDE. The headers of each packet initiated from a server are encrypted with the common key to be identified by each client including unauthenticated ones in the service area. The payload parts of these packets are encrypted with the server key acting as a service key. This key is frequently changed by the system and notification of the change is indicated to the users by the help of index messages in the system. If this message is received by the client, it initiates a new authentication request to get the new service key. The service key is acquired by clients at the end of entity authentication operations.

Entity authentication is accomplished in the controller by comparing the user password encrypted with the client key in the authentication request message with the correspondent in the user authentication table. An unauthenticated client does not know the service key and hence it cannot use received data. Unauthorized access is a possible vulnerability in the system in the cases when the client computer is somehow accessed by other people. To prevent especially this case, the user is forced to provide a username and password in order to be authenticated by the

system. The authentication shall be done in every server change because the subscription of the client to the system may expire after its initial authentication.

Authentication also applies to information itself. We use time stamping for each packet passed between clients and servers. Each party in communication checks the time stamps of the messages, which are received from other parties, and keeps the last received time stamp for each different party. The packets, which have time stamps earlier than or equal to the last encountered time stamp, are discarded. This prevents the situations in which a fake server captures packets and delivers these copies or modified versions of these packets. Additionally, a two-way challenge-response mechanism is applied to all requests and responses. A server announces a challenge for AUP and RQP in start probes. Clients put their responses to that challenge together with their own challenges in the request message. In the notification message, the server sends the response of challenges in the request messages back to clients. This mechanism ensures that the responding party is the one that is expected to respond. A challenge is a random number generated for each CC. There are distinct challenge functions in both clients and servers known by each other. If the response sent to the initiator of a challenge is the same as the result of the challenge function of the responding party, then that packet is recognized as a WIDE packet.

5. Functional Evaluation of the Proposed Mechanisms on a WIDE Prototype

A “proof of concept” prototype of the proposed system is implemented as part of the Mobile Applications and Services Testbed (MAST) [18] of Bogazici University. The prototype provided a testbed for the functional evaluation of the proposed data delivery mechanisms and for further studies. We could not perform large-scale tests on the implemented prototype since we do not have enough number of handsets and people to conduct these tests. In this section, the functionality and behavior of the prototype is presented to observe that the system is functioning

in a stable and expected way with the given experiment setup.

5.1. Implementation and Test Environment of WIDE Prototype

The components are implemented using Microsoft Visual C++ 6.0 and Microsoft Platform Software Development Kit (SDK) for Visual C++ 6.0. The client prototype is designed to run on laptop computers, which have Windows 98 Second Edition or later operating system on them for full functionality. The server and cluster controller prototypes run on desktop computers, which have a Windows 2000 Family operating system on them. In addition, we have a client prototype having partial functionality implemented using .NET Compact Framework to run on a Toshiba E740 PDA with Pocket PC 2002 operating system.

The client application is executed on a laptop computer with a Pentium III processor operating at 500 MHz and 192 MB RAM with Windows 2000 Professional Edition operating system. The server and the controller applications are executed on another desktop computer with a Pentium IV processor operating at 1800 MHz and 1 GB RAM with Windows 2000 Advanced Server Edition operating system.

The wireless connectivity between the server and the client is provided with a Cisco AiroNet 350 Series wireless access point connected to the server computer via an Ethernet adapter and a 3Com AirConnect WLAN PCMCIA card installed on the client.

In our experiments, we set the data payload size to a value of 1400 bytes because the maximum throughput of UDP on the wireless medium is achieved when the UDP packet size is 1472 bytes and throughput rapidly drops after this value [19]. Each data packet contains the header of its own which is added by the server application. The sum of data packet payload, data packet headers and UDP headers should be less than 1472 bytes.

The lengths of the time periods in a CC were also kept fixed in the tests. The durations of

RPP, AUP, and RQP were set to 100 ms, 10 ms, and 10 ms respectively on the server where the timing is achieved with the high-resolution timer provided by the Windows Multimedia API of Platform SDK to provide exact timing at the application layer. The time periods and delay between packets are obtained by waiting between each successive “send” command.

In the experiments, we used files of sizes varying from 100 KB to 1000 KB with an increment of 100 KB. In each test, experiments were repeated at least 15 times and results are represented as their average value. Typically, the number of carousel cycles (nCAR) is set to two. However, in our tests, we set this parameter to one to be able to detect the number of cycles necessary for the completion of the reception of an information service.

5.2. Tuning of Socket Buffer Size and DBDP

The delay between data packets (*DBDP*) parameter is defined as the delay between each data packet of an information service during delivery. In the tests, the client computer is placed indoor, five meters from the WAP, in clear line of sight, to evaluate the effect of buffer size and *DBDP* on the performance. The requested information service was a file of size 100 KB. The socket buffer sizes were varied between the Windows default value of 8 KB to 256 KB increasing in powers of two. In these tests, we measured the time between the reception of the first packet and the last packet of the file at the application layer using the “receive” function of WinSock API. The experiments were repeated for different values of *DBDP* varying between 0 ms to 30 ms with an increment of 10 ms.

The buffer size of the sockets on the client is an important factor that affects the reception time perceived at the application layer as shown in Figure 5. The figure supports the fact that a large buffer is better. However, we should decide on the value for the size of the socket buffer such that there are not any buffer overruns, and the reception time is kept as low as possible.

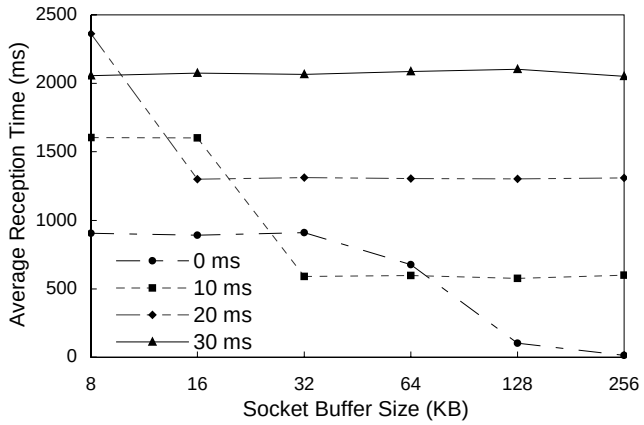

Figure 5. The effect of buffer size and *DBDP* on average reception time

From the figure, we get the idea that the value of DBDP is also an important factor affecting the reception time. As a result of these functional experiments, we decide to select the size of the buffer size to be 256 KB for a file of 100 KB for the following experiments. However, for a file size of 200KB, we cannot achieve the same performance in terms of reception time with *DBDP* set to 0 ms. In this experiment; we fix the buffer size of sockets to 256 KB. In Figure 6, the effect of *DBDP* values on reception time at the application layer is presented. The DBDP values below 3ms cause buffer overruns on the WAP and hence cause the reception time to increase. Hence, we shall decrease the packet arrival rate to the WAP by increasing *DBDP* value.

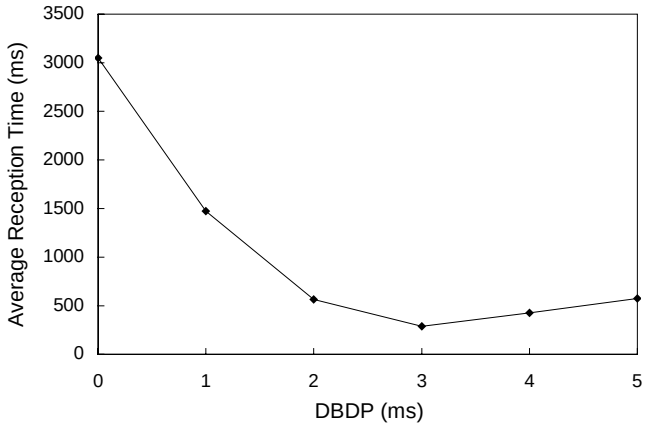

Figure 6. The effect of *DBDP* on average reception time for 200 KB file

In this experiment, we observed that the best performance was achieved when *DBDP* value is around 3 ms which is the selected value of the DBDP in the following experiments. Although this result can also be obtained with a queuing analysis, we obtained this result by actual experiments on the prototype, which presents that the system components are functioning properly. As we compare the results of 3 ms value in Figure 6 and 0 ms values in Figure 5, we conclude that the DBDP is load dependent and higher throughput can be achieved for file sizes around 100 KB by setting smaller DBDP values.

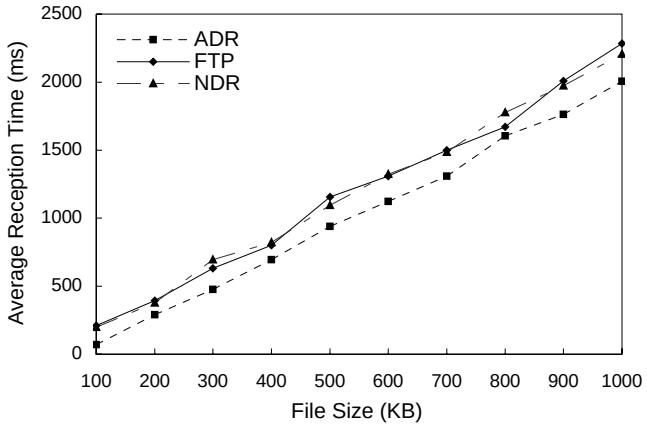

Figure 7. Comparison of average reception time with FTP

5.3. Effect of Information Service Size on Reception Time

We repeated the same experiments for all file sizes between 100 KB and 1000 KB by setting *DBDP* value to 3 ms to see the effect of file size on the reception time. Results of these experiments are presented in Figure 7. In these experiments, we measured the reception time at the application layer perceived by the user (ADR), which is the elapsed time between reading the first packet and the last packet from the socket buffer. For comparison, we measured the elapsed time between the first packet and the last packet captured from the WLAN with the help of the network analyzer [20], which is represented by the network layer data reception (NDR) line. Our aim in this experiment is to compare the performance of the proposed CC structure with an

established protocol, which is FTP. The FTP line represents the average reception times reported by the FTP client.

We observe that NDR performs similar to FTP which shows that UDP achieves TCP-like performance with optimal timing in WIDE. Moreover, in existence of multiple users interested in the same service, the reception time would be the same for all users because of multicasting, in contrast to the FTP case where the medium is shared between all FTP sessions.

In addition, we observe that the reception time in NDR is observed to be greater than the reception time in ADR as shown in Figure 7. The most reasonable cause of this behavior is the context switches to or from the data reception threads forced by the operating system. Because of the latency in reading the packets from the buffers, the packets accumulate in the buffers prior to any read operations. When the context is acquired, packets are read from the buffer for duration of the quantum size of the operating system. Hence, at the application layer, the reading speed is faster than the arrival rate of packets, which explains the time difference between the application layer and network layer reception times, which is approximately 150ms.

5.4. The Effect of Server Load on Reception of Services

In this section, we want to find out DHCP and authentication delays experimentally as well as CC durations, to give an insight about the time that a client needs to download requested service, and therefore to show that the system is able to deliver the service to clients while they are in the coverage which is a design goal stated in Section 2. For this purpose, we configure a large number of carousel cycles so that the server can continuously deliver the same information service of specified size in each CC, which we refer as *load*, with single request to make enough number of measurements during this period. The load varies between no loads to 1000 KB load.

We observe that the average time needed to obtain a valid IP address from the DHCP server when the client enters the service area is found to be 2714.73 ms on the average.

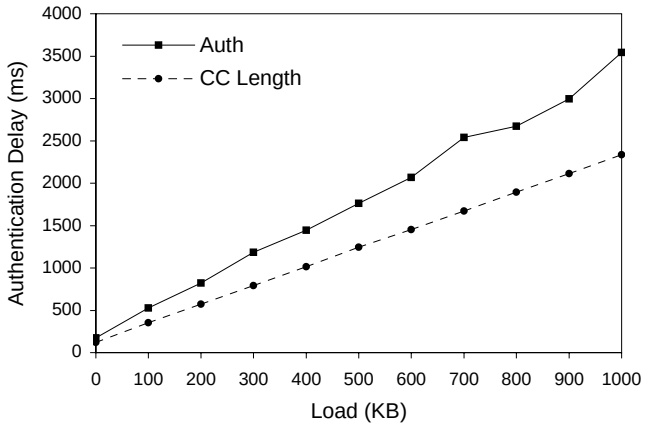

Figure 8. The effect of load on authentication

We measured the time between the end of address configuration and the end of the authentication processes. The results for these tests together with the average duration of the CC for different loads are presented in Figure 8. The increase of the load results with the increase of the length of the CC, which directly increases the authentication delay.

The completion of the address configuration may be at any time point in any time period of CC perceived by the client. In the same way, the authentication responses from the controller may arrive to the server at any time point in any time period of CC. After the address configuration, client has to wait until the arrival of the subsequent authentication start probe to transmit its authentication request to the server. This waiting time lies in the range between zero CC length and one CC length. In addition, client has to wait at least one CC to receive the authentication response. Hence, probabilistically the expected authentication delay is 1.5 times the CC length in the best case, which coincides with the observed result.

Therefore, we conclude that despite such delays, the time a mobile is in the coverage area of a server is sufficient for many communication cycles, which enables the complete reception of

the services, for the assumed walk-through scenarios.

5.5. Mobility and Distance Tests

We designed an experiment to analyze the effect of mobility and distance on the performance of file reception. For this analysis, the client was mobile outdoor in the line of sight. The buffer size was set to 256 KB and *DBDP* value is set to 3 ms. The carousel mechanism is disabled for this test. We also did not employ the FEC mechanism. Tests were done at distances 15, 30, and 50 meters from the WAP. We requested three files of sizes 100 KB, 500 KB, and 1000 KB one at a time at each distance. In the mobility tests the speed of the client was approximately equal to 3 to 4 m/s, i.e., approximately fast walking speed.

Table 1. Average number of cycles for completion of file reception

		Distance		
		15 m	**30 m**	**50 m**
File Size	**100 KB**	1	1	1.2
	500 KB	1	1.2	2.1
	1000 KB	1	1.4	2.3

(a) Client is stationary

		Distance		
		15 m	**30 m**	**50 m**
File Size	**100 KB**	1	1	1.4
	500 KB	1.4	2	2.5
	1000 KB	1.9	2	2.8

(b) Client is moving

In Table 1, average number of cycles needed to complete the data receptions is presented. Table 1.a shows the stationary case and Table 1.b shows the moving case. As expected, the number of packet losses increases as the distance between the WAP and the mobile client increases. In an environment, where FEC mechanism is not employed, any packet loss results in the client waiting for the next cycle to complete the reception in the case of the occurrence of the loss of one of the packets, not received in a previous CC. The increase in the number of packets of an information service increases the probability of occurrence of packet losses and hence, leads to an increase in the average number of cycles for reception.

We can observe that mobility increases the average number of cycles for reception. This was an anticipated result because mobility is known to increase the error rate. However, mobility

does not significantly affect the average number of cycles much for the files of small sizes, because the smaller the number of packets a file has, the smaller is the probability of packet losses. Because of this reason, the mobility did not have too much negative effect on the distance test results presented in Table 1.a.

We repeated the distance and mobility experiments by employing the FEC mechanism with 10% redundancy. In the distance tests, all of the requested information services were received in one cycle. However, when the mobility was included, only 90% of the information services could be received in one cycle. The rest of the services are requested for retransmission and received in the next delivery.

6. Conclusion

The design of an information delivery system, namely WIDE, which targets itinerant clients, is presented in this paper. The system uses the WLAN hot spot concept to disseminate high-bandwidth popular data in a distributed manner. IEEE 802.11b technology is used as the underlying infrastructure for the proposed system. The overall system design is shaped according to some design principles presented in Section 2. The system architecture is constructed for a moderate-size network with the assumption of walk-through scenarios. The communication protocols are presented in view of the communication cycle concept. The mechanisms, which are required for reliable and secure data dissemination, are presented briefly. Proposed data delivery mechanisms are tested on a “proof of concept” prototype and some results on the functional evaluation are reported. Readers interested in the details of the design, implementation and other tests performed on the prototype can refer to the WIDE technical report [21].

Although most of the ideas were theoretically proposed before, it should be noted that the strength of the paper is the presentation of the original overall system design and the actual

implementation of the prototype using IEEE 802.11b technology. The idea in this paper is to analyze the existing solutions for related problems in such systems and combine the solutions in a systematic way in order to come up with a working, real-life, easily deployable prototype and present the functionality of it.

In future studies, our aim is to conduct large-scale performance experiments on the implemented prototype with realistic number of clients and servers and find out how the system behaves under different load conditions. Additionally, we want to modify the design of the system and the implementation of the prototype to improve the power-awareness of the system by dozing the Wi-Fi card at every unnecessary period in the communication cycle. Moreover, we plan to improve the system architecture in terms of scalability and robustness. For this purpose, it is being planned to add backup authentication and profiling services to the current design. With these additions, the system will be able to operate under heavy load conditions without affecting the performance dramatically. We also want to modify the WIDE system to deliver location based information services to its clients. For this purpose we need to find the physical location of the client. WIDE may be integrated with WLAN Tracker [22], but will require modifications to the current design. Currently WIDE offers file delivery services. In the future framework, the system can be improved to give streaming and upload services such as music and video streaming, and e-mail transfer requiring special coding and security issues.

References

1. Frenkiel, F., B. R. Badrinath, J. Borras, and R. Yates, “The Infostations Challenge: Balancing Cost and Ubiquity in Delivering Wireless Data”, *IEEE Personal Communications*, pp. 66-71, April 2000.
2. Borras J., *Capacity of an Infostation System*, Ph.D. Dissertation, Rutgers University, 2000.
3. DATAMAN Laboratory, *NIMBLE: Many-time, Many-where Communication Support for Information Systems in Highly Mobile and Wireless Environments*, http://www.cs.rutgers.edu/dataman/nimble/, 2003.
4. WICAT, Polytechnic University, *Infostation Project*, http://wicat.poly.edu/ infostation.htm, 2003.
5. Frankl P. and D. Goodman, “Delivering Information "To Go" via Infostations”, http://wicat.poly.edu/ tech_report/tr/02-008.pdf.
6. MIND Laboratory, Uni. of Maryland, *Rover Technology*, http://mindlab.umd.edu/research_rover.html, 2003.

7. Bell Labs, *Möbius Project: Scheduling for Large-Scale Data Dissemination Systems*, http://www.bell-labs.com/project/mobius/, 2003.
8. WideRay Project, http://www.wideray.com, 2004.
9. Droms, R., "Dynamic Host Configuration Protocol", *IETF RFC 2131*, March 1997.
10. McAuley, A., S. Das, S. Madhani, S. Baba and Y. Shobatake, *Dynamic Registration and Configuration Protocol*, http://hnmclab.csie.chu.edu.tw/~tmc/sip/draft-itsumo-drcp-01.txt, 28 May 2003.
11. Perkins, C. E., "Mobile IP", *IEEE Communications Magazine*, vol. 3, pp. 84-99, May 1997.
12. Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", *IETF RFC 2462*, December 1998.
13. Imielinski T., S. Viswanathan and B.R. Badrinath, "Energy Efficient Indexing on Air", *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 25-36, 1994.
14. Acharya, S., M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcast", *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 183-194, 1997.
15. Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review*, Vol. 27, No. 2, pp. 24-36, April 1997.
16. Rizzo L. and L. Vicisano, "RMDP: an FEC-based Reliable Multicast Protocol for Wireless Environments", *Mobile Computing and Communications Review*, Vol. 2, No. 2, pp. 1-10, April 1998.
17. Menezes, A. J., P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC press, 1996.
18. Mobile Applications Testbed (MAST), Netlab, Bogazici University, http://netlab.boun.edu.tr/mast.html
19. Vasan, A. and A. U. Shankar. *An Empirical Characterization of Instantaneous Throughput in 802.11b WLANs*, http://www.cs.umd.edu/~shankar/Papers/802-11b-profile-1.pdf, 2003.
20. TamoSoft Inc., CommView 4.0 Evaluation Version, http://www.tamofiles.com/cv4.zip.
21. Donmez, M. Y. and S. Isik, "Design and Implementation of WIDE System", *WIDE System Technical Report*, Bogazici University, TR-023, June 2003.
22. Komar, C. and C. Ersoy, "Location Tracking and Location Based Service Using IEEE 802.11 WLAN Infrastructure", *European Wireless 2004*, Barcelona Spain, 24-27 February 2004.